REMARKS

Claims 1-22 are pending in the present application. Claims 1-22 were rejected. Reconsideration of the claims is respectfully requested.

Applicants thank the Examiner for the teleconfernece conducted on October 6, 2004. During the teleconfernece, the following points were discussed:

I. 35 U.S.C. § 102, Anticipation

The Office Action has rejected claims 1-5, 9-14, 16, and 19-21 under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,974,541 to Hall et al. (hereinafter Hall). This rejection is respectfully traversed.

With respect to this rejection, a prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. In re Lowry, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. Kalman v. Kimberly-Clark Corp., 713 F.2d 760, 218, U.S.P.Q. 781 (Fed. Cir. 1983). In this particular case, each and every feature of the presently claimed invention is not identically shown or described in *Hall*, arranged as they are in the claims.

For example, claim 1 recites the following:

> 1. A method for asynchronous execution within a program, comprising:
> executing code in a first thread;
> determining whether a first keyword exists in the code, the first keyword indicating a code element that may be executed out of order; and
> executing the code element in a second thread.

With regard to claim 1, the Office Action states the following:

> As per claim 1, Hall teaches the invention as claimed, including a method for asynchronous execution within a program, comprising:
> executing code in a first thread (col. 4 lines 1-14; col. 12 lines 24-31);
> determining whether a first keyword exists in the code, the first keyword indicating a code element that may be executed out of order (col. 4 line 57-col. 5 line 8; col. 6 lines 17-33; col. 7 lines 26-30); and

executing the code element in a second thread (col. 8 lines 45-52; col. 13 lines 59-67).

Office Action dated 7/14/2004, pages 2-3.

Applicants respectfully disagree. Hall describes a system and method for asynchronous event notification in a bus system. A notify request that specifies a device to be monitored is provided to a driver level software by an application. The driver level software then monitors the device for events that are to be monitored. When an event being monitored occurs, the driver level software recognizes the event and invokes a call back function of the application. Invocation of the callback function is thus performed asynchronously from the application. That is, the callback function is not invoked by a scheduled mechanism, e.g., a polling function or the like, of the application. Hall in no manner describes, suggests, or otherwise alludes to asynchronous execution of code including "determining whether a first keyword exist in the code" where the keyword indicates "a code element that may be executed out of order," nor for "executing the code element" that is indicated as able to be executed out of order "in a second thread." For example, the passages of Hall cited in the Office Action as describing the claim 1 limitation of determining the existence of a first keyword that indicates a code element may be executed out of order are as follows:

> Synchronous Event Notification
>
> In many GPIB applications, GPIB events are used to monitor system status or respond to instruments that are requesting service. Traditional methods for responding to GPIB events involve *synchronous waits* on the events of interest or polling loops that periodically check the GPIB status variable, ibsta, for the events of interest.
>
> Asynchronous Event Notification
>
> According to the present invention, GPIB *applications can asynchronously receive* event notification using one of two mechanisms, these being the ibnotify function, or the GpibNotify OLE control. Asynchronous callback functions offer a new approach for automatically executing blocks of code when one or more GPIB events are detected. Using the ibnotify, or the GpibNotify ActiveX (OLE), function and the asynchronous event notification technique, applications can eliminate wasted processing time associated with using synchronous waits or the polling technique.

Hall, Column 4, Line 57- Column 5 Line 8 (*emphasis added*).

> A more efficient solution to this problem is to use an asynchronous callback function according to the present invention, which is called immediately upon the occurrence of a GPIB event. Win32 GPIB applications can asynchronously receive event notifications using the ibnotify function or the GpibNotify OLE control. This is useful if the user desires the application to be notified asynchronously about the occurrence of one or more GPIB events. Using ibnotify, or GpibNotify, the application does not need to check the status of the GPIB device. When the GPIB device requests service, the GPIB driver software automatically notifies the application that the event has occurred by invoking a callback function. The callback function executes when any of the GPIB events specified in the event mask parameter have occurred.

Hall, Column 6, Lines 18-33

> The purpose of ibnotify and GPIBNotify is to provide asynchronous event notification, e.g., to notify the user of one or more GPIB events by invoking the user callback. ibnotify is a function, and GpibNotify is an OLE control.

Hall, Column 7, Lines 26-30

Thus, Hall describes a mechanism for asynchronously receiving event notifications by an application. Particularly, Hall describes a callback function that facilitates asynchronously providing event notification to an application - that is, a mechanism that provides event notification without requiring the application to poll or otherwise perform a periodic check for the event. Hall in no manner describes or suggests asynchronous code execution within a program. The passages of Hall in no manner describe a mechanism for "determining whether a first keyword exists in" code that indicates "a code element that may be executed out of order" and is wholly unrelated to asynchronous program execution. Thus, Hall is thoroughly insufficient to anticipate the subject application as claimed in independent claim 1.

As described in the present application, a mechanism is provided for asynchronously executing a statement in a program. A keyword is used to indicate that a statement may be executed asynchronously, i.e., out of order, with respect to other statements at the same nesting level in the code. When a compiler or interpreter encounters a statement that may be executed asynchronously, a thread is then created for execution of the code that may be executed asynchronously. (See Subject Application, Page 3, Lines 1-20; Page 10, Lines 16-22). For example, Figure 4A of the subject application shows the following:

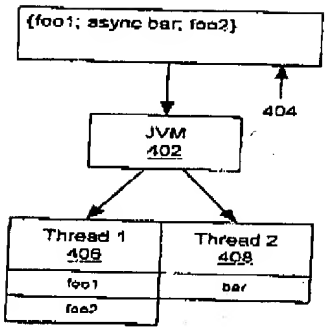


Figure 4A

AUS9-2000-0720-US1
Sheet 4 of 6

As can be seen, a method includes statements and a keyword (async) that designates a statement (bar) that may be executed out of order. Code that is not to be executed out of order is executed in a first thread, and the code that is indicated by the keyword as being able to be executed out of order is executed in a second thread.

Independent claims 10 and 19 recite similar features as claim 1. Therefore, the same distinctions between Hall and the claimed invention in claim 1 apply for these claims. For the reasons described above, Hall does not contain all elements of independent claims 1, 10 and 19. Hence, Hall fails to anticipate the present invention as recited in claims 1, 10 and 19. Since claims 2-5 and 9 depend from claim 1, claims 11-14 and 16 depend from claim 10, and claims 20-21 depend from claim 19, the same distinctions between Hall and the claimed invention in independent claims 1, 10, and 19 apply for these claims. Additionally, claims 2-5, 9, 11-14, 16,and 20-21 claim other additional combinations of features not suggested by Hall. Consequently, it is respectfully urged that the rejection of claims 1-5, 9-14, 16, and 19-21 under 35 U.S.C. § 102(b) as being anticipated by Hall have been overcome, and such a notice is respectfully requested.

II. 35 U.S.C. § 103, Obviousness

The Office Action has rejected claims 6, 15, and 22 under 35 U.S.C. § 103(a) as being unpatentable over Hall in view of U.S. Patent No. 5,551,040 to Blewett (hereinafter Blewett). This rejection is respectfully traversed.

With regard to claim 6, the Office Action states the following:

> Blewett teaches...determining whether a third keyword exists in the code element, the third keyword indicating a statement that may be executed out of order (col. 5 lines 10-26); and
>
> executing the statement in a third thread (col. 5 lines 10-26)
>
> It would have been obvious to one of ordinary skill in the art to combine Hall and Blewett since the asynchronous processing method of Hall is severely limited in that only one asynchronous call may be outstanding at one time (col. 8 lines 1-5). If the asynchronous code block contains another asynchronous call, the call must either be cancelled or the call must wait until the outstanding call is returned. Blewett specifically address this problem by providing a framework to allow nesting of asynchronous callbacks, such that "execution of the callback function for the new signal...may be commenced in exactly the same fashion that invocation of one function by another suspends execution of the invoking function until execution of the invoked function is finished." (col. 5 lines 10-26)

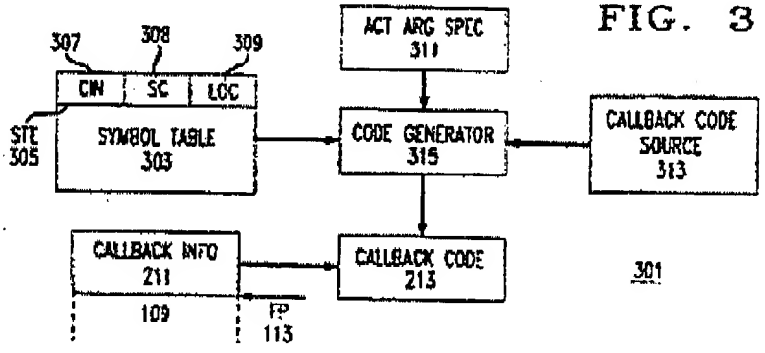
Office Action dated 7/14/2004, Page 7.

Applicant respectfully disagrees. For example, claim 6 recites the following:

> 6. The method of claim 1, further comprising:
>
> determining whether a third keyword exists in the code element, the third keyword indicating a statement that may be executed out of order; and
>
> executing the statement in a third thread.

Blewett describes a mechanism for specifying contexts for executable instructions. Particularly, Blewett describes a mechanism for a callback programming style in which an event handler responds to an event by executing application-level callback code and provides information concerning the event as part of the context of the execution. For example, Figure 3 of Blewett shows the following:

FIG. 3



As can be seen, a code generator receives callback source, an argument specification, and a symbol table to generate callback code. Callback information is placed in the stack such that if another signal (element 204 in Figure 2 of Blewett) occurs while a callback function is being executed, the current execution may be suspended and execution of the callback function for the new signal may commence. The signal is provided by a hardware device, e.g., a mouse (Column 2, Lines 54-56; Column 4, Lines 59-65; and Column 5, Lines 10-26). Blewett neither describes, suggests, or otherwise alludes to asynchronous code execution nor does Blewett describe or suggest a multi-threaded computing environment. Thus, Blewett clearly fails to teach a mechanism for determining whether a "third keyword exists in" the code element that indicates a statement "may be executed out of order," nor for executing "the statement in a third thread." Blewett is unrelated to the teachings of the subject application and is entirely inadequate to provide the deficiencies of Hall. Thus, claim 6 is non-obvious. Moreover, if an independent claim is non-obvious under 35 U.S.C. 103, then any claim depending therefrom is non-obvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Thus, claim 6 is non-obvious as Applicants have already demonstrated claim 1 to be in condition for allowance. Applicants respectfully submit that claim 6 is also allowable, at least by virtue of its dependence on an allowable base claim.

Claims 15 and 22 recite similar features as claim 6. Therefore, the same distinctions between Blewett and the claimed invention in claim 6 apply for these claims. For the reasons described above, Blewett fails to describe the deficiencies of Hall, and neither Hall or Blewett, alone or in combination, contain all elements of claims 6, 15, and 22. Hence, Hall and Blewett fail to obviate the present invention as recited in claims 6, 15, and 22. Consequently, it is respectfully urged that the rejection of claims 6, 15, and 22 under 35 U.S.C. § 103(a) as being obviated by Hall in view of Blewett have been overcome, and such a notice is respectfully requested.

Additionally, the Office Action has rejected claims 7-8 and 17-18 under 35 U.S.C. § 103(a) as being unpatentable over Hall in view of U.S. Patent No. 6,560,626 to Hogle et al. (hereinafter Hogle). This rejection is respectfully traversed.

Hogle describes a thread interrupt mechanism for awakening a blocked thread upon request by another thread. A thread in a waiting or blocked state awaits a condition of the wait state to be satisfied. An interrupting thread terminates the blocked state of the awaiting state so that the interrupting thread can use the function of the blocked thread (Column 2, Lines 13-16; Column 5, Lines 40-51). Hogle in no manner describes or suggests executing a code element out of order by "determining whether a first keyword exists in" code where the keyword indicates "the code element...may be executed out of order." Consequently, Hogle additionally fails to describe or suggest "executing the code element" that has been determined to be able to be executed out of order "in a second thread." Thus, Hogle fails to provide for any of the deficiencies of Hall described above.

Inasmuch as base claim 1 includes elements not shown or described in claims 7 and 8, the same distinctions between Hall and Hogle and the claimed invention in claim 1 apply for these claims. Moreover, if an independent claim is non-obvious under 35 U.S.C. 103, then any claim depending therefrom is non-obvious. Thus, claims 7 and 8 are non-obvious as Applicants have already demonstrated claim 1 to be in condition for allowance. Applicants respectfully submit that claims 7 and 8 are also allowable, at least by virtue of their dependence on an allowable base claim.

With regard to claim 17, Applicants have already demonstrated that Hall fails to describe or suggest "a first keyword indicating a code element that may be executed out of order" and for executing "the code element" in a thread created for the code element determined to be able to be executed out of order. As Hogle fails to describe or suggest a mechanism for executing a code element out of order, Hall and Hogle fail to obviate claim 17. Thus, Applicants submit that claim 17 is allowable. Additionally, claim 18 depends from claim 17, and the same distinctions between Hall and Hogle and the claimed invention in independent claim 17 applies for claim 18. Consequently, Applicants submit that claim 18 is also allowable, at least by virtual of its dependence on an allowable base claim.

Therefore, the rejection of claims 7-8 and 17-18 under 35 U.S.C. § 103(a) as being unpatentable over Hall in view of Hogle have been overcome, and such a notice is respectfully requested.

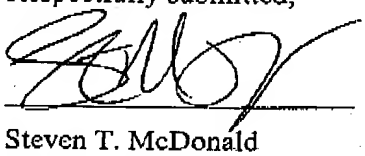
III. Conclusion

It is respectfully urged that the subject application is patentable over Hall, Blewett, and Hogle and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: October 7, 2004

Respectfully submitted,



Steven T. McDonald
Reg. No. 45,999
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Agent for Applicants